

L Number	Hits	Search Text	DB	Time stamp
1	169	"object-oriented" OR "object-relational"	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2002/09/02 09:19
2	8990	"object-oriented" OR "object-relational"	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2002/09/02 09:19
3	891	("object-oriented" OR "object-relational") NEAR5 interface\$2	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2002/09/02 09:19
4	15	((("object-oriented" OR "object-relational") NEAR5 interface\$2) AND (x.500! OR ldap!))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2002/09/02 09:20
5	189	("object-oriented" OR "object-relational") AND (x.500! OR ldap!)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2002/09/02 09:21
6	30	("object-oriented" OR "object-relational") SAME (x.500! OR ldap!)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2002/09/02 09:20
7	6	((("object-oriented" OR "object-relational") AND (x.500! OR ldap!)) NOT @AD>19931230	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2002/09/02 09:22
8	40	("object-oriented" OR "object-relational") SAME "directory service"	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2002/09/02 09:22
9	8	((("object-oriented" OR "object-relational") SAME "directory service") NOT @AD>19931230	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2002/09/02 09:23
10	66	("object-oriented" OR "object-relational") SAME (directory! NEAR2 (database\$2 OR service\$2))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2002/09/02 09:23
11	10	((("object-oriented" OR "object-relational") SAME (directory! NEAR2 (database\$2 OR service\$2))) NOT @AD>19931230	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2002/09/02 09:23
12	25	("object-oriented" OR "object-relational") AND (directory! NEAR2 (database\$2 OR service\$2)) NOT @AD>19931230	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2002/09/02 09:24

[IEEE HOME](#) | [SEARCH IEEE](#) | [SHOP](#) | [WEB ACCOUNT](#) | [CONTACT IEEE](#)[Membership](#) [Publications/Services](#) [Standards](#) [Conferences](#) [Careers/Jobs](#)**IEEE Xplore™**
RELEASE 1.4[Help](#) [FAQ](#) [Terms](#) [IEEE Peer](#) [Quick Links](#) [Review](#)» [Adv](#)

Welcome to IEEE Xplore™

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account

- 1) Enter a single keyword, phrase, or Boolean expression.
Example: acoustic imaging (means acoustic and imaging)
- 2) Limit your search by using search operators and field codes, if desired.
Example: optical <and> (fiber <or> fibre) <in> ti
- 3) Limit the results by selecting Search Options.
- 4) Click Search. See [Search Examples](#)

directory and (relational or sql)

Start Search

Clear

Note: This function returns plural and suffixed forms of the keyword(s).

Search operators: <and> <or> <not> <in> [More](#)Field codes: au (author), ti (title), ab (abstract), ct (conference title), jn (journal name) [More](#)**Search Options:****Select publication types:**

- ☒ Journals
- ☒ Conference proceedings
- ☒ Standards

Select years to search:

From year: 1988 to

Organize search results by

Sort by: Year

In: Descending order

List 15 Results per page

[Home](#) | [Log-out](#) | [Journals](#) | [Conference Proceedings](#) | [Standards](#) | [Search by Author](#) | [Basic Search](#) | [Advanced Search](#)
[Join IEEE](#) | [Web Account](#) | [New this week](#) | [OPAC Linking Information](#) | [Your Feedback](#) | [Technical Support](#) | [Email Alerting](#)
[No Robots Please](#) | [Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)

Copyright © 2002 IEEE — All rights reserved

X.500 Directory and OSI management

Zhang, X. Tesson, J.-L. Seret, D. Remy, C.

Bull S.A., Les Clayes Sous Bois;

This paper appears in: Global Telecommunications Conference, 1992.
Conference Record., GLOBECOM '92. 'Communication for Global Users',
IEEE

12/06/1992 -12/09/1992, 6-9 Dec 1992

Location: Orlando, FL, USA

On page(s): 1106-1110 vol.2

6-9 Dec 1992

References Cited: 12

IEEE Catalog Number: 92CH3130-2

INSPEC Accession Number: 4482776

Abstract:

With the growth of interconnected network size and complexity, diverse requirements have risen for constructing a powerful information service. The concepts of management information base and directory information base are reviewed, and their similarities and differences are analyzed. It is shown that applying X.500 Directory to OSI management presents some interesting benefits. Several X.500 object classes are then proposed to represent management information in the directory. A model of information exchange is addressed briefly. It is suggested that a worldwide information sharing service for management applications can be achieved with the approach described

Index Terms:

OSI management X.500 Directory information exchange information service information services interconnected network management information base open systems telecommunication network management worldwide information sharing service OSI management X.500 Directory information exchange information service information services interconnected network management information base open systems telecommunication network management worldwide information sharing service

Documents that cite this document

Select link to view other documents in the database that cite this one.

X.500 DIRECTORY AND OSI MANAGEMENT

Xinxin ZHANG */**
Dominique SERET **

Jean-Luc TESSERON *
Christian REMY *

* Distributed Computing Operation, Bull S.A.,
Rue Jean Jaures, B.P.68, 78340 Les Clayes sous Bois, France
** Ecole Nationale Supérieure des Télécommunications,
46 rue Barrault, 75634 Paris Cedex 13, France

Abstract

With the growth of interconnected networks in size and in complexity, diverse requirements have risen for constructing a powerful information service. We overview the concepts of Management Information Base and Directory Information Base, and analyse their similarities and differences. Applying X.500 Directory to OSI management presents some interesting benefits. Several X.500 object classes are then proposed to represent management information in the Directory. We also briefly address a model for exchanging information. We believe that, in this way, a world-wide information sharing service for management applications can be achieved.

1. INTRODUCTION

Present-day networks comprise a wide variety of equipment and services offered by different vendors and service providers. The growth in size and in complexity of networks, and especially heterogeneous networks, has spurred the need for powerful integrated network management. Integrated network management aims to provide a single set of tools for managing all the resources within heterogeneous networks, regardless of the type of components. One of the challenges for integrated network management then is how to represent network resources that need to be managed. This is due to the fact that it is used to select managed objects in the exchanges between the management system and managed resources.

X.500 is a set of CCITT recommendations and corresponding ISO standards [CCI-88] for a world-wide directory service. The main objective of the X.500 Directory is to offer the means of determining unknown information from known information. Within heterogeneous networks, information necessary to manage the networks is voluminous -- with the extension of a network to hundreds of thousands computers, locating resources and services becomes very difficult. The X.500 Directory provides a general information sharing service with a refined set of communication protocol facilities and a global name management infrastructure. In addition, the Directory may support a more powerful, more sophisticated set of mechanisms for information look-up in distributed environments. Some publications [LLO-90] [HAL-91] [JAK-90] have dealt with the possible applications of the X.500 Directory in forthcoming broadband networks, mobile communications and OSI systems. In this paper, we intend to envisage the potential to support the management of network services and resources via the X.500 Directory.

This paper is organised as follows. In the ensuing section, we overview briefly the concepts of the Management Information Base (MIB) and the Directory Information Base (DIB). These concepts are compared in Section 3. Using the X.500 Directory functionality for the purpose of network management brings some interesting benefits, which are shortly discussed in Section 4. In Section 5, we address the issue of representing of management information in the Directory. Some new object classes are proposed. On basis of this proposal, an extension of the Directory Information Tree (DIT) structure rules is presented. The aspect of exchanging management information is outlined in Section 6. Our conclusion and areas for further work are presented in the last section of this paper. The work here is very preliminary and does not attempt to propose a complete model, yet it provides a meaningful way to support integrated network management by means of the X.500 Directory. Further research in this area will indicate how this model might be enhanced.

2. OVERVIEW OF INFORMATION STORAGES IN OSI CONTEXTS

In the following, we give a brief overview of the information storage in the OSI systems and in the Directory systems.

2.1. Management Information organisation according to OSI

It has been widely recognised that in order to reduce the complexity of management in heterogeneous networks, we need a global, consistent, and flexible view of network resources. The object-oriented approach is recommended by ISO for representing and organising network resources.

In OSI terminology, a managed object is a management view of a resource or a set of resources, representing only those aspects of the resource(s) that are relevant to the network management. Several types of managed objects are identified by the management standards [ISO-90a]:

- (N)-layer managed object related to a specific layer;
- systems managed object relevant to more than one layer or to the system as a whole;
- physical resources managed object associated to a specific piece of equipment (such as terminal, medium, etc.);
- administrative resource managed object containing administrative or organisational information;

-- management support object which is a systems managed object defined to support a specific management function (e.g. Discriminator, Alarm record); etc.

The "Information Model" [ISO-90b] defines the logical structure of systems management information, which is constructed in terms of managed objects, their attributes, the management operations that can be performed upon them and the notifications that they can emit. It also defines the principles of naming managed objects and attributes so that they may be identified in and accessed by management protocols (CMIP).

The set of managed objects in an open system, together with their attributes, constitutes a logical repository *Management Information Base* (MIB). The MIB must be able to access and change management information in all seven layers of the OSI Basic Reference Model. It is considered as the heart of an integrated management system. The MIB concept does not specify a particular form of physical or logical storage for the information.

The collection of managed objects is arranged hierarchically into a *Management Information Tree* (MIT) or containment tree, which reflects a containment relationship between managed objects and provides a way to specify how to name these managed objects.

Within an OSI system, management activities are structured around the five functional areas: *accounting management, configuration management, fault management, performance management and security management*. Most of the information held within the MIB is shared between two or more functions.

2.2. Information organisation of the X.500 Directory

The X.500 Directory uses information modelling techniques which are closer to the network management standards and object-oriented approach. It holds information on objects and provides users with both retrieve and modify services for accessing the information.

Each object is represented in a Directory as an entry, described by a set of attributes. Each attribute contained in the entry provides a piece of information describing a particular characteristic of the object represented by the entry. Some common types of entries and attributes are standardised. The collection of entries in the Directory, known as the *Directory Information Base* (DIB), is arranged hierarchically into a tree denoted as the *Directory Information Tree* (DIT). Each entry is identified by an unambiguous *Relative Distinguished Name* (RDN). The unique *Distinguished Name* (DN) of an entry is the concatenation of its own RDN and those of all its superiors. An entry's name needs not reflect the physical location of the represented object (the Directory enables users to map user-oriented names onto machine-oriented entries), hence the Directory is independent of the structure of underlying networks.

The Directory adheres to the client-server model. Users interact with a *Directory User Agent* (DUA) which ensures all functions related to the dialogue. The *Directory System Agent* (DSA) performs server functions concerning the access to and the efficient management of information repository. The X.500 Directory is intended to provide a world-wide information sharing service. It is obvious that a centralised directory serving the whole world is not feasible. Apart from the enormous volume of data involved, the number of queries to the Directory might also be overwhelming. Therefore, the Directory may consist of one or more DSAs. In the latter case -- where the Directory is distributed -- each DSA embodies a fragment of the DIT in order to improve

performance and availability. The well-defined *Directory Service Protocol* (DSP) enables communication between DSAs.

3. MIB VERSUS DIB

Hereafter, a short discussion about differences between the MIB and the DIB is given.

3.1. Nature of Information

Managed objects may represent both static and dynamic resources. In essence, the content of a MIB is dynamic, reflecting only the current state of the managed objects.

A DIB might contain some information of a semi-permanent nature, for example personal or organisational information, etc. The information is changed more infrequently as compared with queries. Furthermore, the information in the DIB is updated only through abstract services, i.e. less frequently than in the MIB.

3.2. Database type

The DIB is not a general purpose database, although it may be implemented on such systems. Its organisation is hierarchical. It can function in a distributed environment. In such an environment transient situations can arise where both old and new versions of the same information are available.

The MIB defined by the standards is only an abstract model, which does not address the implementation issue. Methods to choose a database type from the implementation viewpoint have been discussed [BAP-91].

3.3. Object class definition

The managed objects defined by the OSI management standards represent those resource aspects visible to the managing systems. Managed objects with similar characteristics can be grouped into managed object classes. Each managed object is an instance of an object class. An object class may be a subclass of another object class. The templates described in the guidelines for the definition of managed objects [ISO-90c] indicate that a managed object will be also described by its behaviour and the notifications it emits. In contrast, entries in the Directory cannot emit notifications and the standards do not give any information on the behaviour of the entries. There is, however, the same definition on the concept of objects and of the relationship between super-/sub-classes.

3.4. Access services

A standard set of CMIS services [ISO-90d] is provided, which can be used to interact with one or more managed objects in a single request. More complex operations may be performed with the help of a *scope* and a *filter* parameters. Scope defines the depth from a specified base managed object in the MIT, in which the required operation is to be applied. Filter could further be used to extract from the previously scoped managed objects only those which satisfy logical predicates for the operation.

The Directory provides a set of access facilities, known as the Directory abstract services, which allows users to both interrogate the Directory and to modify Directory information. Most of the operations supported by the Directory are aimed at a particular entry only, that is, the name of the entry must be given in the request. Only the *search* abstract service provides similar features as the scope and the filter parameters in the CMIS. The following table gives a more detailed comparison between CMIS operations and the Directory operations.

X.500 Abstract Services		OSI CMISE Services	
Distinguished Name of the Entry must be provided except Search operation	Search Read List	M-Get	associated with scoping/filtering to retrieve any attribute value(s)
Distinguished Name of the entry must be provided. Modify attributes and attribute values. ModifyRDN only applied to leaf entries.	Modify ModifyRDN	M-Set	associated with scoping/filtering. to modify attribute value(s) possible to rename a filtered managed object.
only applied to a leaf entry. all mandatory attributes are to be provided.	Add Entry	M-Create	Attribute value(s) of a reference instance can be default value(s) for new managed object. Name of a new managed object can be determined by the server. Default values can be assigned for mandatory attributes.
Distinguished Name of the entry must be provided only applied to a leaf entry	Remove - Entry	M-Delete	Associated with scoping/filtering; Subordinate managed objects whether or not are removed depending on the request
Distinguished Name of the entry must be provided	Compare	None	
	None	M-Action	to perform an action on managed objects
	None	M-Event -Report	event notification
Search is the only operation giving scoping/filtering	Filtering	Scoping Filtering	more detail than in the X.500, the scope can be defined until level N, or level N only.

3.5. Knowledge

X.500 standards define a model in which the Directory can be distributed over an arbitrary number of DSAs. Each DSA is responsible for a subset of the entries in the DIT. This fragment of the DIT is called the *naming context* of that DSA. The cooperation between DSAs requires that each DSA is aware of the responsibilities of some other DSAs. This is realised by DSAs possessing "knowledge" of the distribution of the Directory. The knowledge information consists of a set of references, such as *internal references*, *subordinate references*, *non-specific subordinate references*, *superior references* and *cross references* etc., which associate DSAs with the fragments for which they are responsible.

The MIB distribution problem was not treated by the management standards. The MIB is limited to one managing system. Each managed object contains the knowledge needed to describe itself for the purposes of management access, for example, managed object name, its object class, etc. There is no knowledge on the schema of the MIB.

3.6. Summary

The MIB -- which represents a data model on complex network entities -- must collect and provide real-time network data to management. The data held by the X.500 Directory is relatively stable. For this reason, the Directory can be used to store data not directly used for monitoring and controlling the network behaviours. The information can be configuration data or statistical data. With the help of the Directory, management applications can apply management activities to a suitable object or support other management functions, for example, fault-, accounting- or configuration-management.

4. BENEFITS OF APPLYING X.500 DIRECTORY TO NETWORK MANAGEMENT

With use of the X.500 Directory functionality, certain management information can be maintained in the Directory to

support management applications. Some benefits from this approach are considered next.

4.1. Use of the X.500 Directory naming

The Directory allows users to define their own objects. The naming architecture is hierarchical and ensures the separation between the names of objects and the structure of the underlying physical networks.

Information on the managed objects can be stored in the Directory. Relationships and associations between managed objects can be defined by means of a naming schema, maintained and managed with a refined set of the Directory protocol facilities and a global name management infrastructure. The naming conventions and associated distributed name resolution procedures standardised in the X.500 Directory could be convenient to use for name look-up in OSI management. The name server function of the X.500 Directory could be used in a distributed environment to retrieve addresses for OSI management applications.

4.2. Use of the X.500 Directory storage facilities

The X.500 Directory could be used as a network information repository to provide a management entity with the necessary information on the environment of a given managed object, such as object definition, functional or administrative data, etc. When managed objects are added to or removed from the network, the data about the changes are collected in the Directory. Thus a dynamic view of the managed object knowledge can be constructed as required. A side-effect is that a subset of data is discharged from the MIB.

4.3. Use of the X.500 Directory "universal access"

The X.500 Directory allows applications and users to locate Directory information over the world. This provides a flexible means to register knowledge information on the MIB schema of a management system. Changes in the definition of managed object classes, object attributes, etc., can be easily marked in the Directory. In this way, facilities for obtaining knowledge in

integrated networks environment are provided to management systems.

4.4. Use of the X.500 Directory security functions

The refined access control model and authentication model standardised in the X.500 Directory could strengthen security aspects of OSI management [ZHA-92]. The security functions of the Directory provides at least twofold capabilities which can satisfy security requirements of management information, such as hiding the contents of some network information from illegitimate users or providing "data privacy" for a selected administrative class of users instead of all users. A more secure way to access and manage the network information can be achieved.

5. MANAGEMENT INFORMATION REPRESENTATION IN X.500 DIRECTORY

Having discussed the feasibility of using the X.500 Directory as an information repository for network management, we now deal with the question of how to represent information necessary to manage network resources in the X.500 Directory.

The network information can include, for example, the name of the managed object, the object class of the managed object, name bindings used to determine the managed objects contained in this managed object, the time when the managed object was created, MIB references for this object, access control information, and also some information on accounting, etc. A collection of object classes and a number of attribute types were defined in X.521 and X.520 respectively. For the purpose of representing management information, we now propose some object classes beyond the scope of the X.500 standards, which are listed in the table below.

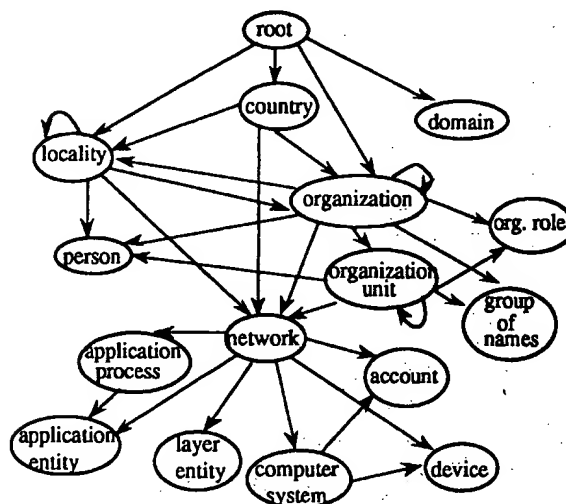
The *Infrastructure* object class defines an object class and attributes to be bound with the entry for a general infrastructure. It can be subclassed so as to define specific systems. The managed object classes (defined by the Forum [FOR-91]), such as Computer System and Network, are subclasses of this object class.

The *LayerEntity* object class reflects the layer management aspect. It may include layer entities, several service access points (SAPs), various communication protocols, etc. An X.25 protocol machine in the network layer is an example of such a resource.

The *domain* object class represents information on domains. Domains play an important role in the design and operation of distributed systems. To define the nature of domains, it must be possible to identify how domains exist, associate and manage the OSI environment and its resources. A mapping of domains onto X.500 systems provides a range of new management and user facilities over and above those currently available.

The *account* object class is used to represent accounting information on network resources. Accounting management is a part of network management which carries out monitoring of the utilisation and charges associated with network components. Adding an object class for accounting is important for allocation and recovery of costs to a directory service as well as a network service.

A very general DIT name binding structure is suggested in Annex B of X.521. The guidelines of X.521 could be followed, but some modifications should be applied to this DIT for management applications, as described below.



The figure above illustrates an example of a DIT name binding structure. Our suggested domain object class is directly subordinated to the root. The Network object class and its associated classes are shown subordinate to organisation class, locality class, country class, etc., but they could equally be placed subordinate to an organisational unit, for instance, in the case of a local network.

In many cases, a network will span several organisations, even a country. We suggest two solutions to the problem of storing network information in such cases, using two facilities defined in the X.500 standards:

-- First, a linkage can be achieved by use of *aliases*. If the network can be regarded as being under the control of a centralised site, a single entry will be used to represent this network; in other organisations alias entries will be created, which may be used to

facilitate searching and to simplify access to the object for collecting the network information.

-- Second, a linkage may be implied by the *seeAlso* attribute. In contrast with the former solution, the network information will be referred to via certain entries in different locations, because each entry may contain an attribute *seeAlso* which gives further search instructions to

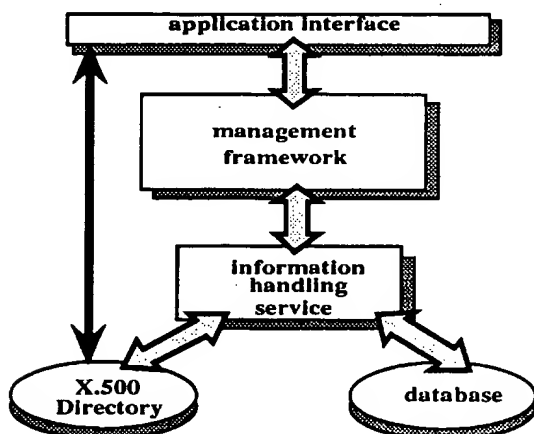
infrastructure	OBJECT-CLASS SUBCLASS OF top
	MUST CONTAIN { commonName }
	MAY CONTAIN { description, manager, locality, seeAlso, owner, provider } :: = { objectClass xx }
layerEntity	OBJECT-CLASS SUBCLASS OF top
	MUST CONTAIN { layerEntityName }
	MAY CONTAIN { layerProtocol, layerServices } :: = { objectClass xx }
domain	OBJECT-CLASS SUBCLASS OF top
	MUST CONTAIN { domainName, domainType, member }
	MAY CONTAIN { manager, description, } :: = { objectClass xx }
account	OBJECT CLASS SUBCLASS OF top
	MUST CONTAIN { commonName }
	MAY CONTAIN { quotas, tariff, usagePattern, description, localityName, organizationName, organizationalUnitName, seeAlso, device } :: = { objectClass xx }

obtain more complete information.

This approach gives a helpful clue to resolving similar problems. In the same way, we could define further "hooking" entries or attributes. Currently, however, it seems sufficient for researching the information.

6. INFORMATION EXCHANGE MODEL

From the above, it is clear that dynamic information about managed objects should be stored in a database as defined by OSI management standards, whilst the configuration or statistical data can be stored in the Directory Information Base. A network information management function is necessary to ensure that information relating to managed objects can be locally or remotely contained, and obtained between management systems. The following figure depicts the architectural relationships between the different components.



management framework The management framework performs all management functions as defined in the OSI systems management standards as well as some non-OSI management functions which are needed for distributed applications. It supports the surveillance and the control of various network components. It receives operation requirements from the interface, interprets them and then dispatches these operations to the desired managed objects with the help of the information handling service for locating the objects.

X.500 Directory The X.500 Directory maintains the information as defined by the X.500 standards as well as the information for network management. It offers the standard functions to register changes in management information, to look up and to deliver the network information under the requests of management applications.

database The database supports the collection, retrieval and manipulation of that management data which is dynamic (i.e. with quickly changing values) and thus cannot be stored using more permanent information storage tools. The relationships attributes in the data [ISO-91] may describe the relationships between the database and X.500 repository. This function is similar to the seeAlso attribute in X.500.

information handling service This module will provide management applications with a consistent interface to access and manipulate data, independent of actual data location or

storage format. Information may be stored in the X.500 Directory or in the database; management applications do not need to know where and how it is stored. When the handling service receives a request from a management application, it consults the X.500 Directory to determine how to communicate with the desired managed object, and to retrieve the information needed to establish communications. If necessary, it binds to the database in order to obtain dynamic management information. The operations offered may include functions to query both the database and the X.500 Directory together in a combined way, and also some more complex functions, such as, a same operation to be provided simultaneously on both the database and the Directory.

7. CONCLUSION

Recently, Directory systems have become a field of interest within the context of open systems. This paper addresses several issues related to applying the X.500 Directory to OSI management. These include a comparison between the MIB for management and the DIB for the Directory services, a specification of how certain management information is incorporated into the underlying data model of the Directory and an information exchange model for ensuring information accessibility to management activities. Some interesting benefits can be reaped through applying the X.500 Directory to management systems.

The work here is very preliminary and the data model proposed is incomplete for actually using the X.500 Directory in the domain of OSI management. Further research work should cover:

- complete definition of the management information representation in the Directory;
- precise definition of the functions of the information handling service module;
- studying the performance of the Directory under the condition that the management applications make use of it; etc.

With more work being done, we believe that more potential applications of the Directory could be discovered.

Acknowledgement

The authors would like to thank F. Klomp, P. Jew and other colleagues for their helpful comments and suggestions.

References

- [BAP-91] S. Bapat : *OSI Management Information Base Implementation*, Proc. of IFIP 2th Inter. Symp. on Intergr. Net. Managt, Washington, April 1991, pp817-832
- [CCI-88] CCITT X500 series/ISO 9594 : *The Directory*, November 1988
- [FOR-91] OSI/NM Forum: *Forum Library of Object Classes, Name Bindings, and Attributes*, Issue 1.2, February 1990
- [HAL-91] J. Hall et al : *Management in a Heterogeneous Broadband Environment*, Proc. of IFIP 2th Inter. Symp. on Intergr. Net. Managt, Washington, April 1991, pp613-624
- [ISO-90a] ISO10040: *Systems Management Overview*, June 1990
- [ISO-90b] ISO10165-1: *Management Information Model*, June 1990
- [ISO-90c] ISO10165-4: *Guidelines for Definition of Managed Objects*, June 1990
- [ISO-90d] ISO9595: *Common Management Information Service Definition*, November 1990
- [ISO-91] ISO10164-3: *Attributes for representing relationships*, August 1991
- [JAK-90] K. Jakobs et al: *Using Directory Services in Mobile Communication*, Proc. of IFIP COMNET'90, Budapest, May 1990, pp219-228
- [LLO-90] A. Lloyd, contribution to ISO/JTC1/SC21: *Modeling OSI Systems Management*, WG4N1029, 12-1-90
- [ZHA-92] X. Zhang, et al: *The Implementation of Access Control in An X500 Environment*, Proc. of the IFIP INDC'92, Helsinki, March 1992, pp154-166



[> home](#) [> about](#) [> feedback](#) [> logout](#)

US Patent & Trademark Office

Citation

ACM Annual Computer Science Conference [>archive](#)
Proceedings of the 1993 ACM conference on Computer science [>toc](#)
1993 , Indianapolis, Indiana, United States

An X.500 prototype to support integrated network management

Authors

Wei Wei
Adrian Tang

Sponsor

ACM : Association for Computing Machinery


Publisher

ACM Press New York, NY, USA

Pages: 251 - 256 Series-Proceeding-Article


Year of Publication: 1993

ISBN:0-89791-558-5

 <http://doi.acm.org/10.1145/170791.170837> (Use this link to Bookmark this page)

[> full text](#) [> abstract](#) [> references](#) [> index terms](#) [> peer to peer](#)

[> Discuss](#) [> Similar](#) [> Review this Article](#)

 [Save to Binder](#)

[> BibTex Format](#)

[↑ FULL TEXT:](#)  [Access Rules](#)

 [pdf 618 KB](#)

[↑ ABSTRACT](#)

An X.500 Prototype to Support Integrated Network Management

Wei Wei and Adrian Tang

OSI Laboratory
Computer Science and Telecommunications Program
University of Missouri at Kansas City
Kansas City, MO 64110

Abstract *Because of the heterogeneity and the distributed nature of integrated network management, a good approach to understand the managed objects by syntax/semantics on the remote managed systems can significantly improve the performance of management applications. In this paper, we describe an X.500 prototype to provide the directory services to support integrated network management. Four directory services are described: partial name resolution, abstract syntax resolution, navigation of Management Information Base, and access control to managed objects.*

1 Introduction

Naming and addressing are fundamental problems in almost all network application systems. There is no exception to network management. In many cases, it is generally assumed that a manager has the knowledge of all the managed objects on the remote managed systems. This assumption not only puts too much burden on the manager, but also makes the integration of network management systems extremely difficult.

There are at least two fundamental problems in integrated network management. One, as mentioned above, is the problem of naming and addressing. In order to access a managed object in a remote managed system, the management operation issued by the manager must specify the distinguished name (DN) of that managed object in the remote *Management Information Base* (MIB). Since there are usually more than one managed system in a network, there are a number of MIBs. Hence to manage a huge network, the manager needs to maintain a directory of the DNs of the managed objects in the different MIBs.

The other fundamental problem is how the manager can understand the semantics of the remote managed objects. Currently, different management profiles use different templates to define managed objects. To support the integration of these management profiles, some dictionary is needed to specify the template used in each management profile, and the managed object classes supported by each profile.

It is the aim of this paper to describe an X.500 prototype to solve the above problems. In Section 2, an overview of the OSI network management model is given. In Section 3, the directory object classes to support integrated network management are defined. Finally in Section 4, the directory services to support network management are discussed.

2 An Overview of the OSI Management Model

The OSI standard [3] uses an object-oriented approach to specify the managed resources. Managed Objects are the management "view" of these resources. An MIB is an organization of the managed objects. The management functions on the MIB are performed by an agent upon receipt of directives specifying the management operations on the managed objects [4].

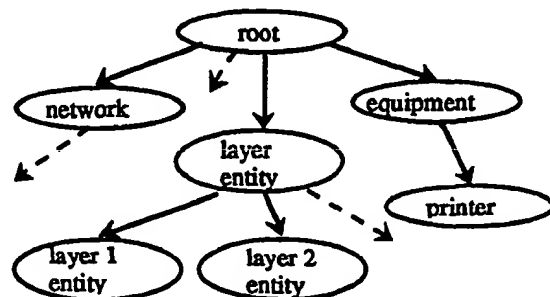


Figure 1 An Example of an Inheritance Tree

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1993 ACM 0-89791-558-5/93/0200/0251 \$1.50

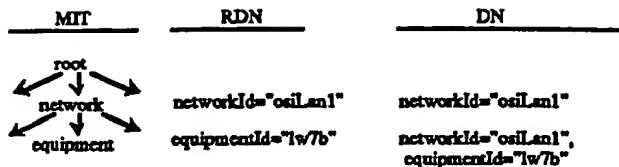


Figure 2 Part of an MIT

A managed object can be described in terms of attributes, notifications which it may emit, and the operations that can be performed upon the object or its attributes. Managed objects with similar characteristics can be grouped into a managed object class. New classes can be derived from existing classes, where a new class inherits the characteristics of one or more parent classes. This inheritance property exploits the commonality among objects and holds the key to upward compatibility. An inheritance tree, built using the inheritance property, is given in Figure 1. In other cases, a managed object can be regarded as contained within another managed object (e.g. a network connection within an Ethernet card). Such a containment relation is used to define a *Management Information Tree* (MIT) which provides names to managed objects (Figure 2). Each managed object in the MIT has its local name, known as a *Relative Distinguished Name* (RDN). The DN of a managed object is obtained by concatenating in a sequence the RDNs of all its superiors in the MIT.

```

equipment MANAGED OBJECT CLASS
  DERIVED FROM {top}
  CHARACTERIZED BY equipment-Package
  REGISTERED AS { obj-class}
          
```

```

equipment PACKAGE
  BEHAVIOUR DEFINITION equipment-behaviour
  ATTRIBUTES equipmentId GET,
    administrativeState GET-REPLACE,
    operationalState GET,
    usageState GET,
    managementState GET,
    locationName GET-REPLACE,
    contactNames ADD-REMOVE,
    equipmentPurpose GET-REPLACE,
    vendorName GET-REPLACE,
    userFriendlyLabel GET-REPLACE;
  NOTIFICATIONS objectCreation,
    objectDeletion,
    attributeValueChange,
    stateChange,
    environmentalAlarm,
    equipmentAlarm;
  REGISTERED AS { package };
          
```

Figure 3 Managed Object Class Definition for Equipment

Figure 3 gives the definition of the 'equipment' managed object class, and Figure 4 the 'printer' managed object class. Note that a package is used in the definition to group related attributes.

```

printer MANAGED OBJECT CLASS
  DERIVED FROM {equipment}
  CHARACTERIZED BY printer-Package
  REGISTERED AS { obj-class}

printer PACKAGE
  BEHAVIOUR DEFINITION printer-behaviour
  ATTRIBUTES printerType GET,
    printingSpeed GET,
    fonts GET-REPLACE,
    pitch GET,
    .....
  REGISTERED AS { package };
          
```

Figure 4 Managed Object Class Definition for Printer

Figure 5 describes an instance of 'printer' managed object class. It is a representation of a laser printer in our OSI Laboratory.

```

class = 'printer'
equipmentId = 'LW7B'
operationalState = 'In-service'
location = 'OSILan1'
userFriendlyLabel = 'osiLaser'
printerType = 'laserWriter Plus'
.....
          
```

Figure 5 An Instance of Printer

3 Directory Object Classes to Support Integrated Network Management

The OSI Directory standard suggests a DIT structure [1]. Based on the suggested structure, an extended DIT structure (Figure 6) is proposed in this section to accommodate the management requirements.

The relevant directory object classes in the extended DIT structure can be classified into two categories. The first category contains object classes that serve as an Encyclopedia for network management. The object classes here are the MIS Library class, the managed object syntax class, the package syntax class, and the managed attribute syntax class. For instance, a manager can interrogate the managed attribute syntax class to find out the abstract syntax of a particular managed object class. The second category contains information specific to an agent. The object classes in this category are the supported managed object class and the supported managed object class. For instance, a manager can interrogate the supported managed object class to find out what kind of managed object classes are supported by a particular agent.

Next, we explain the object classes in each category in detail.

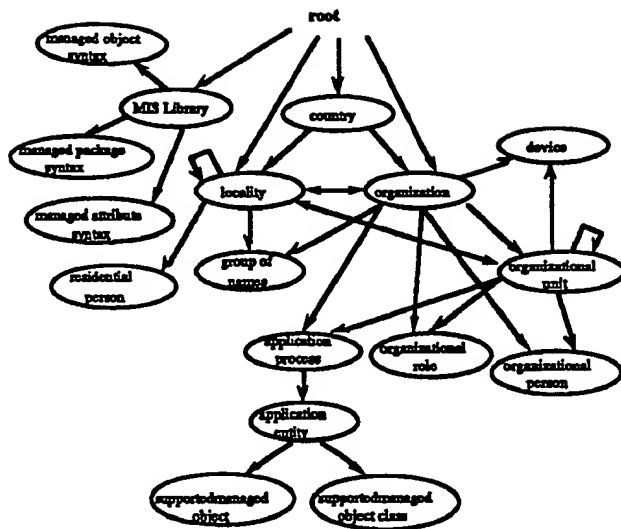


Figure 6 Extended Suggested DIT Structure

The *MISLibrary* directory object class (Figure 7), which is derived from top, has only one mandatory attribute, namely, template. This attribute is used to identify the profile group which is responsible for the definition of the managed object class. Its value, for example, can be CNMA, NMSIG, TIM1, etc.

The *managedObjectSyntax* directory object class (Figure 7) defines the mapping between a managed object class type and its abstract syntax. It has five mandatory attributes and six optional attributes. The mandatory attributes are:

1. *managedObjectClass* attribute which identifies the managed object class,
2. *managedObjectClassId* which gives the object identifier of the managed object class,
3. *derivedFrom* which gives the parent object class(es),
4. *packageList* which gives the mandatory management packages, and
5. *conditionalPackageList* which gives the conditional packages.

The optional attributes are:

1. *description* attribute which gives a description of the managed object class,
2. *seeAlso* which is used to reference some other relevant directory object class,
3. *attributeList* which defines the mandatory attributes,
4. *conditionalAttributeList* which gives the conditional attributes,
5. *operations* which defines the operations permitted to be performed on the managed object class, and
6. *notifications* which defines the notifications that may be emitted by the managed object class.

```
misLibrary OBJECT-CLASS
  SUBCLASS OF top
  MUST CONTAIN { template }
  MAY CONTAIN { description }
  ::= { misLibrary-oid }
```

```
managedObjectSyntax OBJECT-CLASS
  SUBCLASS OF top
  MUST CONTAIN { managedObjectClass,
    managedObjectClassId,
    derivedFrom,
    packageList,
    conditionalPackageList }
  MAY CONTAIN { description,
    seeAlso,
    attributeList,
    conditionalAttributeList,
    operations }
  ::= { m-object-syntax-oid }
```

```
managedPackageSyntax OBJECT-CLASS
  SUBCLASS OF top
  MUST CONTAIN { packageType,
    packageId,
    attributeList,
    behaviourDefinition,
    notificationList }
  MAY CONTAIN { description,
    accessList }
  ::= { m-package-syntax-oid }
```

```
managedAttributeSyntax OBJECT-CLASS
  SUBCLASS OF top
  MUST CONTAIN { managedAttributeType,
    managedAttributeId,
    attributeSyntax }
  MAY CONTAIN { description,
    groupAttribute,
    actionList }
  ::= { m-attribute-syntax-oid }
```

Figure 7 *MIS Library* Directory Object Classes

The *managedPackageSyntax* directory object class (Figure 7) defines the mapping between a package type and its abstract syntax. It contains five mandatory attributes: the *packageType* attribute which gives the package name, the *packageId* which identifies the package, the *attributeList* which gives the attributes that are used to compose the package, and the *notificationList* which gives those notifications that can be emitted by the managed object class. The managed package syntax object class contains two optional attributes. For example, the *accessList* attribute is used to limit accessibility to the package, depending on the granularity of the security.

For an example of the above directory object classes, we consider the system managed object class defined by the CNMA profile. The values of the managed object syntax class and the managed package syntax class are given in Figure 8 and Figure 9 respectively.

```

class = 'managedObjectSyntax'
managedObjectClass = 'system'
managedObjectClassId = 'cnma function
managedObjectClass (3) 2'
derivedFrom = {top}
packageList = {system-package}
conditionalPackageList = {systemTime-package,
stationRestart-package}

```

Figure 8 An Instance of the Directory managedObjectSyntax Class

The *managedAttributeSyntax* directory object class (Figure 7) defines the mapping between an attribute type and its syntax. It has three mandatory attributes: the *managedAttributeType* which gives the managed attribute name, the *managedAttributeId* which identifies the attribute, and the *attributeSyntax* which defines the syntax of the attribute.

```

class = 'managedPackageSyntax'
packageType = 'system-package'
packageId = 'cnma function package (4) 81'
attributeList = {(systemProfile, (GET)),
(managerName, (GET-REPLACE,
ADD-REMOVE)),
(geographicalLocation, (GET-REPLACE)),
(sampleTime, (GET-REPLACE))}
notificationList = NULL

```

Figure 9 An Instance of the Directory managedPackageSyntax Class

The *supportedManagedObject* directory object class (Figure 10) defines the scheme for friendly naming resolution for management system. It has three mandatory attributes: *commonName* which gives a user friendly name of the managed object, *managedObjectClassId* which gives the object identifier of the managed object, and *mibDN* which gives the distinguished name of the managed object in an agent's MIB.

```

supportedManagedObject OBJECT-CLASS
SUBCLASS OF top
MUST CONTAIN { commonName,
managedObjectClassId,
mibDN }
MAY CONTAIN { description,
seeAlso,
accessList }
::= { objectClass xxx }

```

Figure 10 Supported Managed Object Object Class

The *supportedManagedObjectClass* directory object class (Figure 11) defines the managed object classes supported by an agent. The *superiorList* attribute gives the superiors of the managed object class in the agent's MIB, the *subordinateList* attribute gives the subordinates of the

managed object class in the agent's MIB, and the *accessList* sets the constraints on the accessibility to the managed object class.

```

supportManagedObjectClass OBJECT-CLASS
SUBCLASS OF top
MUST CONTAIN { managedObjectClass,
managedObjectClassId }
MAY CONTAIN { superiorList,
subordinateList,
accessList }
::= { objectClass xxx }

```

Figure 11 Supported Managed Object Class Object Class

To illustrate the use of the above object classes, we consider the 'printer' object used in Figure 4. Figures 12 and 13 give the corresponding values of the *supportedManagedObject* and *supportedManagedObjectClass* for the 'printer' object.

```

CLASS = 'supportedManagedObject'
commonName = 'OSI laser'
ID = 'xxx'
mibDN = { networkId='osiLan1',
equipmentId='1w7b' }
.....

```

Figure 12 'OSI laser' Directory Object Instance

```

CLASS = 'supportedManagedObjectClass'
managedObjectClass = 'printer'
ID = 'yyy'
superiorList = 'network'
.....

```

Figure 13 'printer' Directory Object Instance

4 Directory Services to Support Integrated Network Management

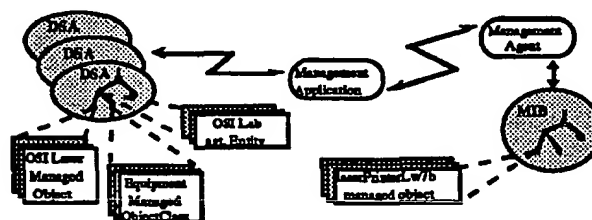


Figure 14 Interactive Model of Directory and Management

In this section, we discuss how to use the directory services to support network management. In our discussion, we use our OSI laboratory as the environment. For simplicity sake, we assume that our Laboratory supports only one managed object, namely, an OSI Laser

Printer. On the DSA side, the DIB (Figure 14) is established following the guidelines of the extended suggested DIT structure (Figure 6). The OSI Lab agent entity is a refinement of the application entity object in the DIB. The supportedManagedObjectClass object class contains only one managed object class, namely, the Equipment object class. The supportedManagedObject object class contains only one managed object, namely, the OSI LASER PRINTER. Four of the possible directory services to support network management are:

1. partial name and management environment resolution,
2. abstract syntax resolution,
3. navigation of the MIB infrastructure, and
4. access control to the managed objects.

In the following, we will describe each one of the above services.

4.1 Partial Name Resolution

By using our extended directory model, the manager not only can retrieve the presentation address of a remote agent and its associated management protocol(s), but also can retrieve the DN of a remote managed object. After learning the management protocol used by the remote agent and the DN of a managed object local to that agent, the manager can then set up an application association with the remote agent using the appropriate management protocol and manipulates the remote managed object using its DN. Figure 15 shows the set_operation for which the manager only has to provide a user friendly name instead of the DN of the remote managed object. The mapping of user friendly names to DNs is performed by some directory interrogation service.

```
set_operation(MO_user_friendly, agent_process_title) {
    use directory interrogation with agent_process_title as
    parameter to find out the presentation address and the
    management protocol used by the agent process;

    use directory interrogation with MO_user_friendly name
    as parameter to find out the DN of the managed object
    in the remote agent;

    case management_protocol:
        CMIP: M_SET(MO_MIB_DN, attr., value, .....);
        SNMP: SET(.....);
        .....
}
```

Figure 15 An Implementation of the Management Set operation

4.2 Abstract Syntax Resolution

The OSI Directory can be used as an information database to provide the abstract syntax information of a given

managed object. There are currently many different management profiles involved in defining managed objects. Each profile uses its own template. It will take some time for these profiles to converge to use the same template. For the time being, in the absence of a common template, the manager needs to understand the syntax and semantics of a remote managed object before it can perform any management operation on the managed object.

Extending the DIT to include the MISLibrary object class provides a convenient on-line abstract syntax look-up for a managed object and its related characteristics. We allow more than one management template to co-exist. The template attribute of the MISLibrary object class describes the template as used by a management profile.

4.3 MIB Navigation

Management information browser is an important application in network management. It enables the manager to browse through the MIT as well as the MIB of a remote managed system without any hard-wired knowledge of the management environment in the target managed system. Our X.500 prototype supports browsing through the managed object classes of a remote agent via the supportedManagedObjectClass object class.

However, browsing through an MIB instance can only be partially done, because some managed objects (such as alarms, logs, and processes), due to their dynamic nature, cannot be registered with the DIB. Their registration with DIB would require extensive updates to the directory objects to keep them consistent with the corresponding MIB.

The approach suggested in [6] is very generic. It always starts the browse at an instance of the class *system*, which is assumed to be the top object in every managed system. Its method to get all the attributes of a managed object by specifying the no attribute list in the M-GET operation. Also by using scoping, one can find the subordinates of an object.

While [6] uses local databases to store the mappings between the class, attributes and event identifiers and their names and abstract syntaxes as defined by the OSI management standard, we introduce the MIS Library object in our X.500 prototype to function as an encyclopedia to store these mappings for (OSI or non-OSI) standards as well as profiles. Another important difference between [6] and our approach is that our browse can start at any node at remote MIB, because the partial name resolution service can return the DN for that node.

4.4 Access Control to MIB

An MIB usually contains a lot of sensitive information. If access control is performed by a managed agent, it would put too much burden on the agent in a large scale environment.

On the other hand, X.500 not only defines an authentication framework for OSI applications but also defines the Basic Access Control Scheme which controls access to the user/operational information in the DIB. This Basic Access Control Scheme defines the protected items,

the user classes, the permission categories required to perform each directory operation, and the Access Control Decision Function to decide whether a requestor can be granted a permission.

Our X.500 prototype introduces the accessList attributes in the supportedManagedObjectClass and the supportedManagedObject object class. The former implements access rules through roles, and the later refines the access rules for individual managed objects.

```
accessList ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX
    AccessControlList
  ::= { attributeType xxx }

AccessControlList ::= SET OF
  SEQUENCE OF {
    operation (Identifier of operation)
    SEQUENCE OF {
      CHOICE {
        distinguishedName
        filter } } }
```

Figure 16 Definition of the accessList attribute

The accessList attribute is given in Figure 16. Its semantics is that each manager, which is described by a name or a filter, is authorized to perform an indicated management operation.

5. Conclusion

In this paper, we propose an X.500 prototype, along with four possible directory services, to support integrated network management. MIS Library and related object classes are proposed to provide syntactic/semantic information of remote managed objects for different network management standards and profiles. Based on this new X.500 prototype, we propose directory services to support partial name resolution, abstract syntax resolution, navigation of remote MIB, and access control to managed objects.

References

- [1] ISO/IEC 9594-6: Information Technology - Open Systems Interconnection - The Directory - Part 6: Selected Attribute Types.
- [2] ISO/IEC 9594-7: Information Technology - Open Systems Interconnection - The Directory - Part 7: Selected Object Classes.
- [3] ISO/IEC 10165-4: Information Technology - Open Systems Interconnection - Structure of Management Information - Part 4: Guideline for the Definition of Managed Objects.
- [4] Adrain Tang, Sophia Scoggins, "Open Networking with OSI", May 1992, Prentice Hall.
- [5] ISO/IEC 8824: Specification of Abstract Syntax Notation One (ASN.1)
- [6] G. Pavlou, J. Cowan and J. Crowcroft, "A General Management Information Base Browser". Proceedings of the 1992 IFIP International Conference on ULPA, May 27-29, 1992, Vancouver, Canada.

[IEEE HOME](#) | [SEARCH IEEE](#) | [SHOP](#) | [WEB ACCOUNT](#) | [CONTACT IEEE](#)[Membership](#) [Publications/Services](#) [Standards](#) [Conferences](#) [Careers/Jobs](#)**IEEE Xplore®**
RELEASE 1.4[Help](#) [FAQ](#) [Terms](#) [IEEE Peer](#) [Quick Links](#)» [Advanced Search](#)[Review](#)

Welcome to IEEE Xplore®

- ☐ [Home](#)
- ☐ [What Can I Access?](#)
- ☐ [Log-out](#)

Tables of Contents

- ☐ [Journals & Magazines](#)
- ☐ [Conference Proceedings](#)
- ☐ [Standards](#)

Search

- ☐ [By Author](#)
- ☐ [Basic](#)
- ☐ [Advanced](#)

Member Services

- ☐ [Join IEEE](#)
- ☐ [Establish IEEE Web Account](#)
- ☐ [Access the IEEE Member Digital Library](#)

1) Enter a single keyword, phrase, or Boolean expression.
Example: acoustic imaging (means the phrase acoustic imaging plus any stem variations)

2) Limit your search by using search operators and field codes, if desired.

Example: optical <and> (fiber <or> fibre) <in> ti

3) Limit the results by selecting Search Options.

4) Click Search. See [Search Examples](#)

Note: This function returns plural and suffixed forms of the keyword(s).

Search operators: <and> <or> <not> <in> [More](#)

Field codes: au (author), ti (title), ab (abstract), jn (publication name), de (index term) [More](#)

Search Options:

Select publication types:

- ☒ IEEE Journals
- ☒ IEE Journals
- ☒ IEEE Conference proceedings
- ☒ IEE Conference proceedings
- ☒ IEEE Standards

Select years to search:

From year: to

Organize search results by

Sort by:

In: order

List Results per page

[Home](#) | [Log-out](#) | [Journals](#) | [Conference Proceedings](#) | [Standards](#) | [Search by Author](#) | [Basic Search](#) | [Advanced Search](#)
[Join IEEE](#) | [Web Account](#) | [New this week](#) | [OPAC Linking Information](#) | [Your Feedback](#) | [Technical Support](#) | [Email Alerting](#) | [No Robots Please](#) |
[Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)

TDB-ACC-NO: NN91021

DISCLOSURE TITLE: Address Book Class Hierarchy.

PUBLICATION-DATA: IBM Technical Disclosure Bulletin, February 1991, US

VOLUME NUMBER: 33

ISSUE NUMBER: 9

PAGE NUMBER: 1 - 4

PUBLICATION-DATE: February 1, 1991 (19910201)

CROSS REFERENCE: 0018-8689-33-9-1

DISCLOSURE TEXT:

- Provided is class hierarchy of objects for an object-oriented design which will maximize code reuse and allow considerable flexibility in developing advanced address book-related functions. It overcomes the current difficulties in application development by building a set of classes providing basic address book behaviors within individual classes. Each class is responsible for its own methods. In addition, this address book design allows for items other than people to provide addressing information. In the following design, a distribution list can cause distribution to a printer, a file folder and other objects in addition to providing access to the addresses of people.

- By subclassing the provided classes, application developers can quickly produce an improved version of the address book application and add new function. The amount of code to be added or changed is minimized by the encapsulation of the classes. Existing code is easily reused.

- The problem being addressed by this design is how to build an object-oriented address book which will provide flexibility for an electronic office system, such as distribution to electronic addresses including printers, files, fax machines, telephones and to people, as well. Current non-object-oriented architectures limit mail distribution to mailboxes for people.

- The proposed class hierarchy consists of three categories of objects. Classes are divided into model and view objects in a revised Model-View-Controller scheme first published by ParkPlace Systems in Smalltalk 80. In addition, database objects have been added.

- The model objects consist of the data portion of the object and its behaviors. The classes in the design which are models are AddressBook, AddressBookContainer, and the subclasses of AddressBookContainer which are ConferenceRooms, DistributionLists, People, PersonalEntries, Organizations, Printers, and Services. The AddressBook class hierarchy for the model classes is shown in the top half of Fig. 1. The data objects themselves which are the address book records for individual items are shown in the lower half of Fig. 1. For example, Person is a record for a particular person; OfficePrinter is a record for a particular printer.

- The view objects are the window-related objects which dictate how the models are displayed and how the end-user interacts with them. The view objects in this scheme are subclasses of a general ViewHandler class which defines the parts each view has: a window, an icon, etc. There is a one-to-one relationship between models and view handlers. Each model has a corresponding view handler so there is an AddressBookViewHandler, an AddressBookContainerViewHandler and

view handlers for all the other AddressBookContainer subclasses such as ConferenceRoomsViewHandler.

- The database classes are the classes which interface with the relational database where the addressing information is stored. The data for each model object being contained is derived from a table in the database. For example, a PersonalEntry is retrieved from the table OffPersEntries. A class OffPersEntries is defined in the class hierarchy. It retrieves and stores data in the OffPersEntries table. The database class hierarchy is shown in Fig. 2.

- Since a significant number of the AddressBook classes are containers, there is also a containment hierarchy which describes the types of objects contained, i.e., the 'contains' relationship, and this is shown in Fig. 3.

- The advantages of the Address Book Class hierarchy are:
production of reusable code for addressing and
distribution of information
increased programmer efficiency
modularity of design
localization and encapsulation of addressing function
provision for electronic distribution to objects other
than people.

SECURITY: Use, copying and distribution of this data is subject to the restrictions in the Agreement For IBM TDB Database and Related Computer Databases. Unpublished - all rights reserved under the Copyright Laws of the United States. Contains confidential commercial information of IBM exempt from FOIA disclosure per 5 U.S.C. 552(b)(4) and protected under the Trade Secrets Act, 18 U.S.C. 1905.

COPYRIGHT STATEMENT: The text of this article is Copyrighted (c) IBM Corporation 1991. All rights reserved.

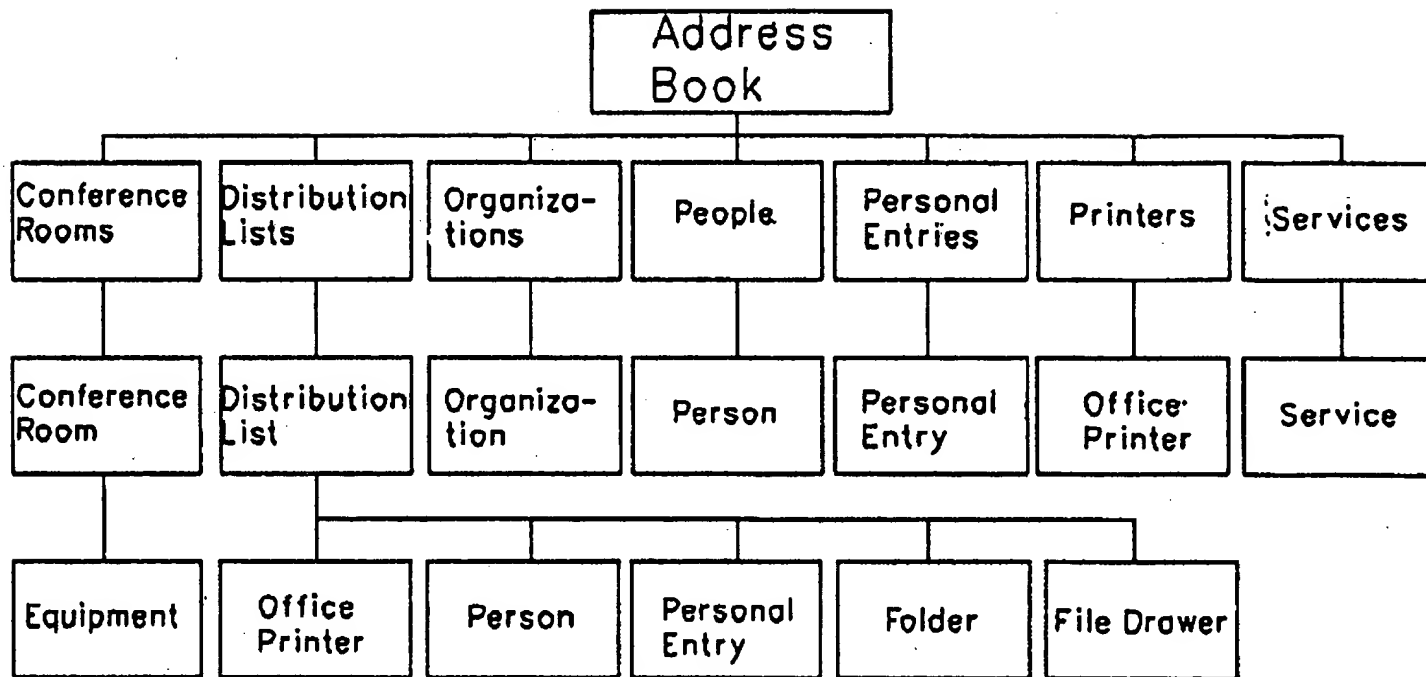


FIGURE 3

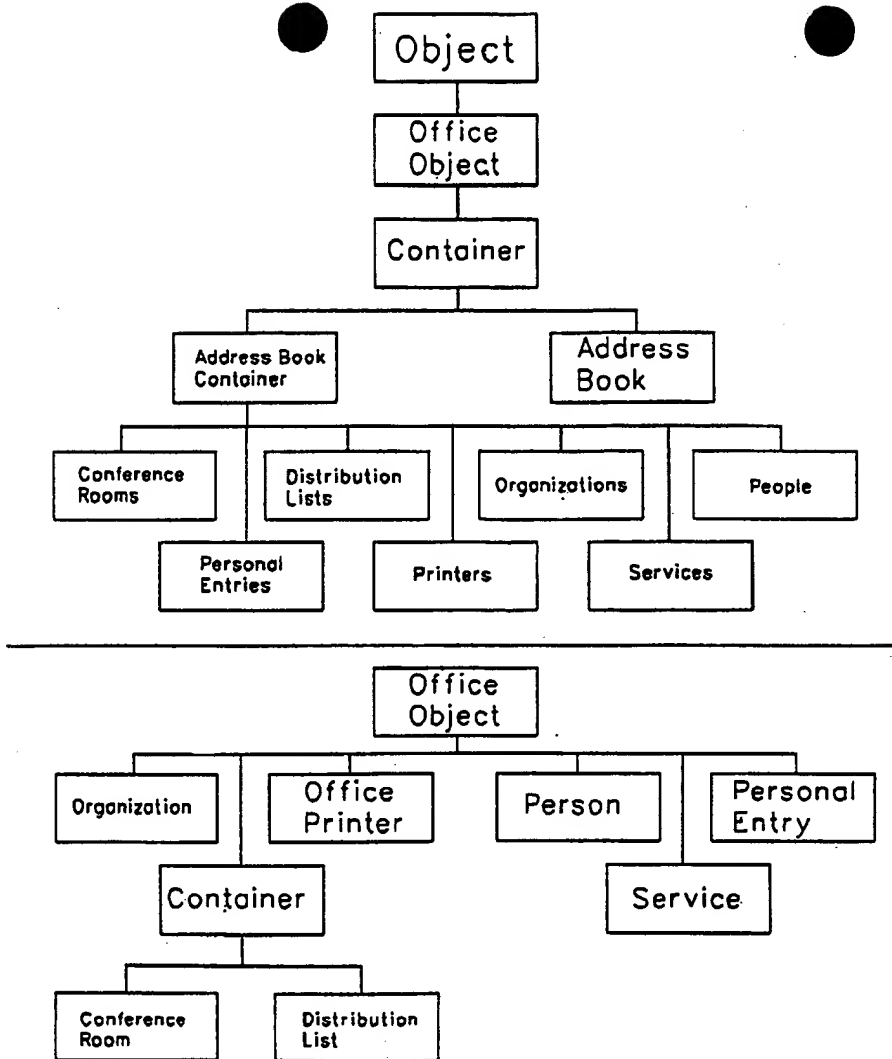


FIGURE 1

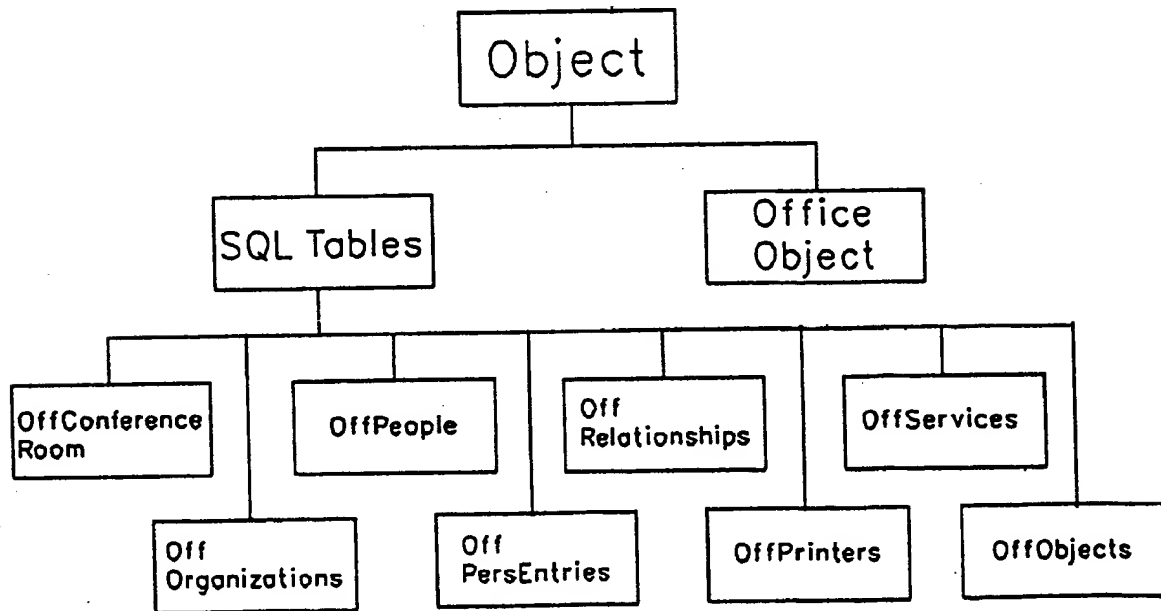


FIGURE 2

US-PAT-NO: 6052681
DOCUMENT-IDENTIFIER: US 6052681 A
TITLE: X.500 system and methods

----- KWIC -----

Harvey; Richard Hans

In order to do comparisons (e.g. search for a particular value), the syntax rules can be applied to create a normalized form (e.g. "CHRIS MASTERS"). If this normalized form is stored in the database, then any variations in input form are effectively removed, and exact matching can be used (which is necessary when using SQL).

Both the normalized data and "raw" data are stored in the database. The "raw" data is necessary so that users can retrieve the data in exactly the same format as it was originally input. As per the X.500 and LDAP standard, data received from a user, raw data, accords with ASN.1. (Abstract Syntax Notation No. 1). Thus the "Name" column in the Hierarchy Table becomes the "NameRaw" and a "NameNorm" column is added. Similarly, the "Value" column in the Object Table becomes the "ValueRaw" and a "ValueNorm" column is added.

Any data supplied by an X.500 service is supplied as a list of ObjectId's and their associated values. These must be converted into AID's (using the Attribute table) and normalized values (using the Object table) for use by the X.500 application. The database returns data s AID's and Raw Values, which must then be converted into ObjectId's and their associated values in the X.500 result.

Compare returns a `matched` or `not matched` result. A raw value is input but the compare is performed using the normalized value.

a storage arrangement operative to concurrently store information in the database in both a protocol encoded form and a syntax-normalized form.

8. A database application as claimed in claim 4, wherein the meta-data is stored such that it exists concurrently in both a protocol encoded form and a syntax-normalized form.

a) converting an incoming X.500 services request into at least one query on tables with syntax-normalized data, and

15. An apparatus as claimed in claim 14, wherein tables containing syntax-normalized data are clustered around an attribute identifier.

19. A method as claimed in claim 18, where the step of applying a filter is applied only to syntax-normalized meta data.

29. A method of implementing X.500 services, is claimed in claim 26, 27 or 28, wherein at least one of the steps of applying a process of functional decomposition, service decomposition, and physical transformation results in data being stored concurrently in a protocol encoded form and a syntax-normalized form.

US-PAT-NO: 6052681
DOCUMENT-IDENTIFIER: US 6052681 A
TITLE: X.500 system and methods

----- KWIC -----

Harvey; Richard Hans

a storage arrangement operative to concurrently store information in the database in both a protocol encoded form and a syntax-normalized form.

3. Apparatus as claimed in claims 1 or 2, wherein the protocol encoded form is ASN.1.

8. A database application as claimed in claim 4, wherein the meta-data is stored such that it exists concurrently in both a protocol encoded form and a syntax-normalized form.

b) basing outgoing X.500 services responses on at least one query on tables with protocol encoded data.

16. An apparatus as claimed in claim 14 or 15, wherein tables containing protocol encoded data are clustered around an entry identifier.

29. A method of implementing X.500 services, is claimed in claim 26, 27 or 28, wherein at least one of the steps of applying a process of functional decomposition, service decomposition, and physical transformation results in data being stored concurrently in a protocol encoded form and a syntax-normalized form.

[> home](#) [> about](#) [> feedback](#) [> logout](#)

US Patent & Trademark Office

Search Results

Search Results for: [(relational AND x.500 AND directory)<AND>(meta_published_date <= 12-01-1993)]
Found 33 of 100,930 searched. → Rerun within the Portal

Search within Results

[> Advanced Search](#) [> Search Help/Tips](#)

Sort by: Title Publication Publication Date Score Binder

Results 1 - 20 of 33 **short listing**

◀
Prev
Page

1

2

▶
Next
Page

-
- | | | |
|----------|---|------------|
| 1 | Nomenclator descriptive query optimization for large X.500 environments
Joann J. Ordille , Barton P. Miller
ACM SIGCOMM Computer Communication Review , Proceedings of the conference on Communications architecture & protocols August 1991
Volume 21 Issue 4 | 94% |
| 2 | Partitioning in X.500
John Henshaw , Michael Bauer
Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: states of the art and practice March 1993 | 91% |
| 3 | ASCW: an assistant for cooperative work
Thomas Kreifelts , Wolfgang Prinz
Proceedings of the conference on Organizational computing systems
December 1993 | 89% |
| 4 | Descriptive names in X.500
G. W. Neufeld
ACM SIGCOMM Computer Communication Review , Symposium | 87% |

proceedings on Communications architectures & protocols August 1989

Volume 19 Issue 4

This paper presents a new name form for the OSI X.500 directory system. The primary function of the directory is to provide a name-to-object look-up facility for OSI objects. The directory consists of a set of agents and a database which is distributed among the agents. The directory database is structured as a tree where each node, or entry, corresponds to an object. An entry consists of a set of attributes where one or more attributes are designated as the object name relative to the entr ...

- 5** Network information centers (NICs): is there one on your horizon? 85%



Susan Calcari , Priscilla Huston , Jan Eveleth , Linda Hutchison , Jerry Martin

Proceedings of the 21st ACM SIGUCCS conference on User services November 1993

- 6** Distributed search for cooperative applications 85%



Michael A. Bauer , Richard A. McBride , J. Michael-Bennett

Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: states of the art and practice March 1993

- 7** Uniform access to Internet directory services 83%



D. Comer , R. E. Droms

ACM SIGCOMM Computer Communication Review , Proceedings of the ACM symposium on Communications architectures & protocols August 1990

Volume 20 Issue 4

As networks and internetworks of computers expand in size and scope, discovery and location of resources becomes a primary function of the networked computing environment. Static tables describing network resources have been replaced by dynamic directory services, such as X.500 and the Internet Domain Name System. These dynamic directory services provide more timely and accurate information about network resources than static tables. A wide variety of services address various com ...

- 8** The profile naming service 83%



Larry L. Peterson

ACM Transactions on Computer Systems (TOCS) November 1988
Volume 6 Issue 4

Profile is a descriptive naming service used to identify users and

encompassing claims. . . . [D]espite extensive statements in the specification concerning all the analogs of the EPO gene that can be made, there is little enabling disclosure of particular analogs and how to make them. Details for preparing only a few EPO analog genes are disclosed. . . . This disclosure might well justify a generic claim encompassing these and similar analogs, but it represents inadequate support for Amgen's desire to claim all EPO gene analogs. There may be many other genetic sequences that code for EPO-type products. Amgen has told how to make and use only a few of them and is therefore not entitled to claim all of them.

927 F.2d at 1213-14, 18 USPQ2d at 1027. However, when claims are directed to any purified and isolated DNA sequence encoding a specifically named protein where the protein has a specifically identified sequence, a rejection of the claims as broader than the enabling disclosure is generally not appropriate because one skilled in the art could readily determine any one of the claimed embodiments.

See also *In re Wright*, 999 F.2d 1557, 1562, 27 USPQ2d 1510, 1513 (Fed. Cir. 1993) (The evidence did not show that a skilled artisan would have been able to carry out the steps required to practice the full scope of claims which encompass "any and all live, non-pathogenic vaccines, and processes for making such vaccines, which elicit immunoprotective activity in any animal toward any RNA virus." (original emphasis)); *In re Goodman*, 11 F.3d 1046, 1052, 29 USPQ2d 2010, 2015 (Fed. Cir. 1993) (The specification did not enable the broad scope of the claims for producing mammalian peptides in plant cells because the specification contained only an example of producing gamma-interferon in a dicot species, and there was evidence that extensive experimentation would have been required for encoding mammalian peptide into a monocot plant at the time of filing); *In re Fisher*, 427 F.2d 833, 839, 166 USPQ 18, 24 (CCPA 1970) (Where applicant claimed a composition suitable for the treatment of arthritis having a potency of "at least" a particular value, the court held that the claim was not commensurate in scope with the enabling disclosure because the disclosure was not enabling for compositions having a slightly higher potency. Simply because applicant was the first to achieve a composition beyond a particular threshold potency did not justify or support a claim that would dominate every composition that exceeded that threshold value.); *In re Vaeck*, 947 F.2d 488, 495, 20 USPQ2d 1438, 1444 (Fed. Cir. 1991) (Given the relatively incomplete understanding in the biotechnological field involved, and the lack of a reasonable correlation between the narrow disclosure in the specification and the broad scope of protection sought in the claims, a rejection under 35 U.S.C. 112, first paragraph for lack of enablement was appropriate.).

If a rejection is made based on the view that the enablement is not commensurate in scope with the claim, the examiner should identify the subject matter that is considered to be enabled.

2164.08(a) Single Means Claim

A single means claim, i.e., where a means recitation does not appear in combination with another recited element of means, is subject to an undue breadth rejection under 35 U.S.C. 112, first paragraph. *In re Hyatt*, 708 F.2d 712, 714-715, 218 USPQ 195, 197 (Fed. Cir. 1983) (A single means claim which covered every conceivable means for achieving the stated purpose was held nonenabling for the scope of the claim because the specification disclosed at most only those means known to the inventor.). When claims depend on a recited property, a fact situation comparable to *Hyatt* is possible, where the claim covers every conceivable structure (means) for achieving the stated property (result) while the specification discloses at most only those known to the inventor.

2164.08(b) Inoperative Subject Matter

The presence of inoperative embodiments within the scope of a claim does not necessarily render a claim nonenabled. The standard is whether a skilled person could